(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 089 188 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.04.2001 Bulletin 2001/14

(51) Int Cl.[7]: **G06F 13/38**

(21) Application number: **00308580.0**

(22) Date of filing: **29.09.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **01.10.1999 US 410565**

(71) Applicant: **Hewlett-Packard Company**
Palo Alto, CA 94304 (US)

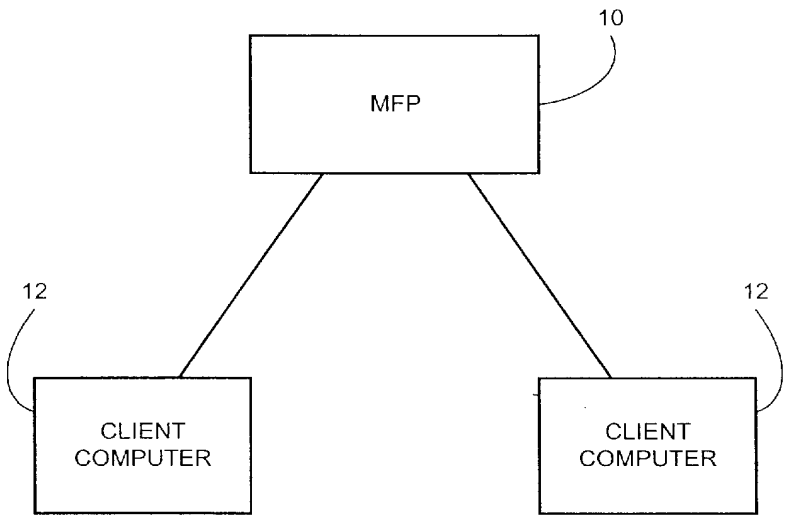
(72) Inventors:
- **Blackwood, Craig M.**
  La Jolla, California 92038 (US)
- **Kumpf, David A.**
  Rocklin, California 95765 (US)
- **Ramchandran, Padmanabhan**
  San Diego, California 92129 (US)

(74) Representative: **Powell, Stephen David et al**
WILLIAMS, POWELL & ASSOCIATES
4 St Paul's Churchyard
London EC4M 8AY (GB)

(54) Event messages communiction between a client and a peripheral device in a computer network

(57) A method for communicating an event message between at least one client (12) operatively connected to a peripheral device (10), the peripheral device being adapted to execute an event when selected by a user or a client, includes maintaining a table of event descriptors in the peripheral device (58, 60, 62, 64, 68, 70, 72, 74), selectively adding entries to the table in response to receiving a preselected add event descriptor request from a connected client (14, 16, 18, 20, 22, 24, 26, 28, 30, 34), selectively deleting entries from the table in response to receiving a preselected delete event descriptor request from a connected client (38, 40, 42, 44, 46, 48, 50, 52, 54, 56), and the user selecting an entry from said table of event descriptors for executing the selected event (84, 90, 92, 94).

FIG. 1



EP 1 089 188 A1

Description

[0001] The present invention generally relates to a method for communicating an event message between one or more client computers operatively connected to a peripheral device, and more particularly to a method for adapting the peripheral device to execute an event when selected by a client computer or a user.

[0002] It is common for a number of client computers to be connected to more than one peripheral, such as a printer and/or a scanner. This arrangement allows a number of client computers to share the services of the printer and/or the scanner. Such client computers can be personal computers or other computers and will be referred to herein simply as a client. It is becoming more popular to use a multifunction peripheral (MFP) to combine the functions of these peripherals into a single peripheral. As a result, the use of the MFP devices are becoming more sophisticated, creating the need for more efficient communications between the MFP device and the clients.

[0003] One prior method of input peripheral event communication involved polling and sending simple button press notifications. However, polling was inefficient because it used the communication bandwidth constantly rather than only when needed. There was also a tradeoff between responsiveness and increased bandwidth use because a prompt response to the user would require frequent polling, which made other communication, such as scanning or printing, take longer.

[0004] Another approach has involved adapting the input peripheral to allow the user to select the destination, and then sending the input data directly to that client computer. Another problem involved with this approach was that the client computer often needed to perform various setup steps before it was ready to receive the data. Furthermore, it often needed additional user input to route the data to the desired destination. Although the input could be performed at the client computer, the user was still required to input data in two different places. Alternatively, the input could be performed at the peripheral, but this required the peripheral to know all possible actions that the client computer might want to take with the data, which made it very difficult to add new functions to client computer software because of the necessity to change the peripheral at the same time.

[0005] Accordingly, it is an object of the present invention to provide an improved method for more efficient communications of an event message between a peripheral device and a client connected to the network.

[0006] It is a further object of the present invention to provide an improved method in which the user is able to complete the entire input task at the peripheral device.

[0007] The present invention provides a method for communicating an event message between one or more clients operatively connected to a peripheral device, and more particularly to a method for adapting the peripheral device to execute an event selected by a client or a user. The user can complete the entire input task at the peripheral device.

[0008] In accordance with this invention, there is provided a method for maintaining a table of event descriptors in the peripheral device, selectively adding entries to the table in response to receiving a preselected add event descriptor request from a connected client, selectively deleting entries from the table in response to receiving a preselected delete event descriptor request from a connected client, and consisting of a user selecting an entry from said table of event descriptors for executing the selected event.

[0009] Other objects, features and advantages will become apparent upon reading the following detailed description, in conjunction with the attached drawings, in which:

FIGURE 1 is a schematic diagram of an input peripheral/client network architecture in which the present method is implemented;

FIG. 2 is a flowchart illustrating the method for selectively adding entries to the table of event descriptors step of the present invention;

FIG. 3 is a flowchart illustrating the method for selectively deleting entries from the table of event descriptors step of the present invention;

FIG. 4 is a flowchart illustrating the method for the client to download entries from the table of event descriptors;

FIG. 5 is a flowchart illustrating the method for the client to upload entries to the table of event descriptors; and,

FIG. 4 is a flowchart illustrating the method in which a user can select an entry from the table of event descriptors of the present invention.

[0010] Broadly stated, the present invention is directed to a method and system for communicating event messages between at least one client operatively connected to a peripheral device. The method allows the user to complete the entire input task at the peripheral device. The peripheral device is adapted to maintain a table of event descriptors, and is also adapted to add and delete entries in the table in response to a preselected event descriptor request from a connected client. In addition, the peripheral device is adapted to allow a connected client to download and upload entries onto the table. Lastly, the peripheral device is adapted to allow a user to select an entry from the table for executing the selected event.

[0011] Turning now to FIG. 1, a schematic diagram of an input peripheral/client network architecture is shown, and illustrates the way in which the client computers are connected for the implementation of the present invention. A peripheral device, such as a multifunction peripheral (MFP) 10, may be connected to a plurality of clients 12 (two shown) to exchange communications of event

messages. The MFP 10 preferably maintains a table of event descriptors. This table of event descriptors is a list of entries from the clients 12, with each entry comprising a string of characters that constitutes a list of parameters that the clients 12 need to execute the event. Each parameter, separated with a semi-colon character, is constructed as a keyword that names the parameter and an associated value defining the parameter. For example, an entry may have the following format: "KEYWORD1=value1; KEYWORD2=value2;". This format gives the clients 12 maximum flexibility because parameters of any length may be introduced without affecting the peripheral device. The only requirement is that the total string not exceed some reasonable maximum length set by the peripheral device.

[0012] FIG. 2 shows the steps involved in which the MFP 10 provides the clients 12 a way to add an entry to the table. A user can send input material from the peripheral by having the client 12 add an entry to the table. For example, one way that the client 12 can add an entry is by using a software program located on the client. The client 12 will first construct an event descriptor string with all the information needed to complete an event message to be included in an add-event message (block 14). For example, the string might specify that the event is a scan and that the input data should be delivered to a word processor located on client 12. The client 12 then sends this add-event message to the MFP 10, requesting that the event be added (block 16). One way to send the add-event message, for example, is by using the Peripheral Management Language (PML) protocol to SET an ADD-ENTRY management object to direct the MFP 10 to add the entry to the table. It should be noted that this task may be accomplished by other protocols. It should also be noted that if there are more than one client 12 connected to the MFP 10, the MFP 10 must then keep track of which client 12 supplied each entry in order to send the add-event message back. After the MFP 10 received the add-event message from the client 12 (block 18), the MFP 10 checks for an available space in the table (block 20). If no available space exists in the table, the MFP 10 will send an error message to the client 12 (block 22), and concludes the subroutine (block 24). When a space is available in the table, the MFP 10 will try to determine whether there is a duplicate entry by trying to find the exact event descriptor string in the table (block 26). If a duplicate is not found, then the MFP 10 adds the new entry to the table (block 28). The MFP 10 next sends a success message to the client 12 as an indicator to the client of the successful completion of the input add-event message (block 30). This concludes the subroutine of adding entries to the table, and the MFP 10 returns to an idle state and waits for the next event message (block 32). On the other hand, an error message will be sent to the client 12 when a space is not available (block 34), and concludes the subroutine (block 36).

[0013] Similarly, FIG. 3 shows the steps involved in which the MFP 10 provides the clients 12 a way to delete an entry from the table. When a user wants to terminate the use of a peripheral or wants to shut down the client 12 so that it can no longer process events, the client 12 will remove the entry or entries from the table. The client 12 will first construct an event descriptor string with the exact same string it added earlier to be included in an delete-event message (block 38). The client 12 then sends this delete-event message to the MFP 10, requesting that the event be deleted (block 40). This again can be accomplished, for example, by using the PML protocol to SET a DELETE-ENTRY management object directing the MFP 10 to delete the entry from the table. After the MFP 10 received the delete-event message from client 12 (block 42), the MFP 10 then tries to locate a match of both the string and the client 12 in the table (block 44). The MFP 10 determines if such a match was found (block 46). The MFP 10 accordingly removes the entry from the table if a match was found (block 48), and may then send a success message to client 12 as an indicator to the client of the successful completion of the input delete-event message (block 50). Again, the MFP 10 returns to an idle state and waits for the next event message since the subroutine of deleting entries from the table comes to a conclusion after the success message is sent (block 52). On the other hand, if the MFP 10 was unable to find a match in the table, an error message is sent to the client 12 (block 54). An additional step for the client 12 to display the error message to the user may be included to keep the user informed of the status (block 56).

[0014] FIG. 4 shows the steps involved in which the MFP 10 provides the clients 12 a way to download an entry or entries from the table. The client 12 first sends a download-request message to the MFP 10 (block 58). MFP 10 receives this download-request message (block 60), and accordingly sends the event descriptor strings for the requested download entries to the client 12 (block 62). After client 12 receives the download event descriptor strings (block 64), the subroutine is concluded (block 66).

[0015] FIG. 5 similarly shows the steps of the MFP 10 providing the clients 12 a way to upload an entry or entries to the table. This time, the client 12 builds the event descriptor string or strings for the upload request entry or entries, and the string is to be included in a downloaded-request message (block 68). The client next sends this upload-request message to the MFP 10 (block 70). The MFP replaces the table entries with the new event descriptor string as requested (block 72), and concludes this subroutine (block 74).

[0016] In accordance with another aspect of the present invention, FIG. 6 shows the steps involved in which the MFP 10 provides a user a way to select an entry from the table. The MFP 10 waits for the user to start the selection event process (block 76), which takes place when the user invokes an event selection mechanism on the MFP 10. For example, a scanner or a scan-

capable MFP 10 might provide a "SCAN TO" button. When the user presses the button, the MFP 10 selects one entry from the table for display to the user. As a convenience to the user, it is more desirable to select the most recently used entry since this entry would most likely be the one that the user seeks. For example, each time the "SCAN TO" button is pressed, the MFP 10 advances to the next entry in the table and displays it to the user. In this case, the MFP 10 shows the "name" portion of the "DISPLAY=name" parameter in the string (block 78).

[0017] The client 12 then provides a way for the user to choose or input a name that easily identifies the entry to the user, and waits for an input from the user (block 80). The user may be given three choices for the input, specifically choices to move to next entry, select current entry, or cancel. If the user picks the choice to move to next entry, the MFP 10 shows the newly selected entry using the "DISPLAY=name" parameter in the string (block 82) and waits for a new user input (block 80). If the user chooses the input of cancel, the MFP 10 ignores the selection event process initiated earlier by the user (block 86), and concludes the select entry subroutine (block 88). When the user finally selects the entry, the MFP 10 sends the selection-event message with the complete event descriptor string necessary to execute the event back to the client 12 (block 90). The MFP 10 stores this entry as the most recently used entry as an update for next time (block 92). The client 12 receives and acknowledges the selection-event message as required by the underlying messaging protocol, such as the PML protocol. In the case of the PML protocol, a PML TRAP-REPLY message is sent by the client 12. The client 12 then executes the selection-event message, specifically executing the action indicated by the complete event string that was added to the selection-event message (block 66). After the execution, the subroutine of selecting the entry is completed (block 68).

[0018] From the foregoing description, it should be understood that an improved method for communicating event messages between at least client operatively connected a peripheral device has been shown and described which has many desirable attributes and advantages. The method adapts an input peripheral device to enable the user in completing the entire input task at the peripheral device, rather than dividing parts of the task between the peripheral device and the clients.

[0019] While various embodiments of the present invention has been shown and described, it should be understood that other modifications, substitutions and alternatives are apparent to one of ordinary skill in the art. Such modifications, substitutions and alternatives can be made without departing from the scope of the invention, which should be determined from the appended claims.

[0020] Various features of the invention are set forth in the appended claims.

Claims

1. A method for communicating an event message between at least one client (12) operatively connected to a peripheral device (10), the peripheral device being adapted to execute an event when selected by a client or a user, said method comprising the steps of:

   maintaining a table of event descriptors in the peripheral device (58, 60, 62, 64, 68, 70, 72, 74);
   selectively adding entries to said table of event descriptors in response to receiving a preselected add event descriptor request from a connected client (14, 16, 18, 20, 22, 24, 26, 28, 30, 34);
   selectively deleting entries from said table of event descriptors in response to receiving a preselected delete event descriptor request from a connected client (38, 40, 42, 44, 46, 48, 50, 52, 54, 56); and,
   a user selecting an entry from said table of event descriptors for executing the selected event (84, 90, 92, 94).

2. The method according to claim 1 wherein each entry contained in said table of event descriptors is a string of characters constituting a list of parameters needed for at least one client to execute the event message, each parameter is constructed as a keyword that names the parameter and an associated value defining the parameter, and each parameter is separated with a semi-colon character.

3. The method according to claim 1 wherein said selectively adding entries step further comprises the steps of:

   the client constructing an event descriptor string including all the information needed to complete the event message for an add-event message (14);
   the client sending said add-event message to the peripheral device (16);
   the peripheral device receiving said add-event message (18);
   the peripheral device sending an error message to client if the table is full (20, 22); and,
   the peripheral device locating the exact event descriptor string in the table of event descriptors if the table is not full (20, 26).

4. The method according to claim 1 wherein said selectively adding entries step further comprises the steps of:

   the client sending a download-request mes-

sage to peripheral device (58);
the peripheral device receiving said download-request message (60);
the peripheral device sending the download event descriptor strings with the requested download entries from said table of event descriptor to client (62); and,
the client receiving the downloaded event descriptor strings (64).

5. The method according to claim 1 wherein said selectively adding entries step further comprises the steps of:

the client builds event descriptor string for an upload-request message (68);
the client sends said upload-request message to peripheral device (70); and,
the peripheral device replacing event descriptor strings in said table of event descriptor with the requested download-request entries (72).

6. The method according to claim 1, wherein said selectively deleting entries step further comprises steps of:

the client constructing an event descriptor string including the exact event string added earlier for a delete-event message (38);
the client sending said delete-event message to the peripheral device (40);
the peripheral device receiving said delete-event message (42);
the peripheral device locating the exact event descriptor string in said table of event descriptors (44);
the peripheral device removing the entry from said table of event descriptors if said exact event string is found (46, 48); and,
the peripheral device sending an error message to the client if said exact event string is not found (46, 54).

7. The method according to claim 1 wherein said selectively deleting entries step further comprises the steps of:

the client sending a download-request message to peripheral device (58);
the peripheral device receiving said download-request message (60);
the peripheral device sending the download event descriptor strings with the requested download entries from said table of event descriptor to client (62); and,
the client receiving the downloaded event descriptor strings (64).

8. The method according to claim 1, wherein said selectively deleting entries step further comprises steps of:

the client builds event descriptor string for an upload-request message (68);
the client sends said upload-request message to peripheral device (70); and,
the peripheral device replacing event descriptor strings in said table of event descriptor with the requested download-request entries (72).

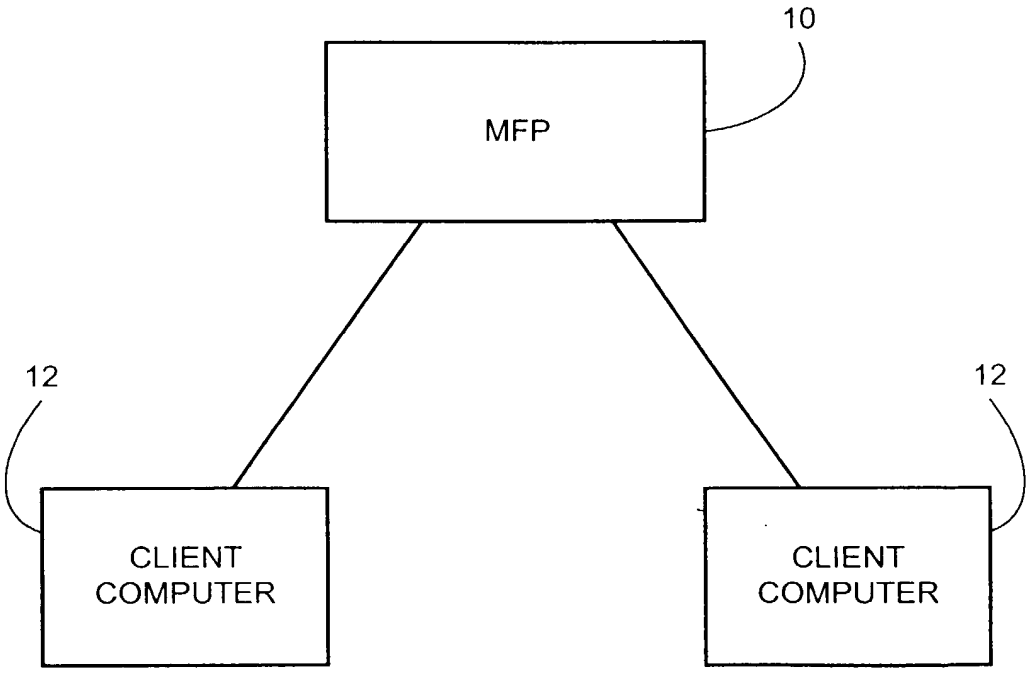
9. The method according to claim 1, wherein said step of a user selecting an entry further comprises the steps of:

the user initiating a selection event process (76);
the peripheral device displaying a most recently used name in the "DISPLAY=name" parameter contained within the entry descriptor string (78);
the client providing a means to allow users to input a name of the entry (80); and,
the peripheral device responding to said input name of the entry from the client (84).

10. A system for enabling the execution of events by a peripheral device, wherein the events are defined by selective event messages and wherein event messages can be requested by a user and at least one client operatively connected to the peripheral device, said system comprising:

means for maintaining a table of event descriptors in the peripheral device (58, 60, 62, 64, 68, 70, 72, 74);
means in the connected client for selectively generating preselected add event descriptor requests for adding entries to said table of event descriptors (14, 16, 18, 20, 22, 24, 26, 28, 30, 34);
means in the connected client for selectively generating preselected delete event descriptor requests for deleting entries from said table of event descriptors (38, 40, 42, 44, 46, 48, 50, 52, 54, 56); and,
a user selecting an entry from said table of event descriptors for executing the selected event (84, 90, 92, 94).

FIG. 1



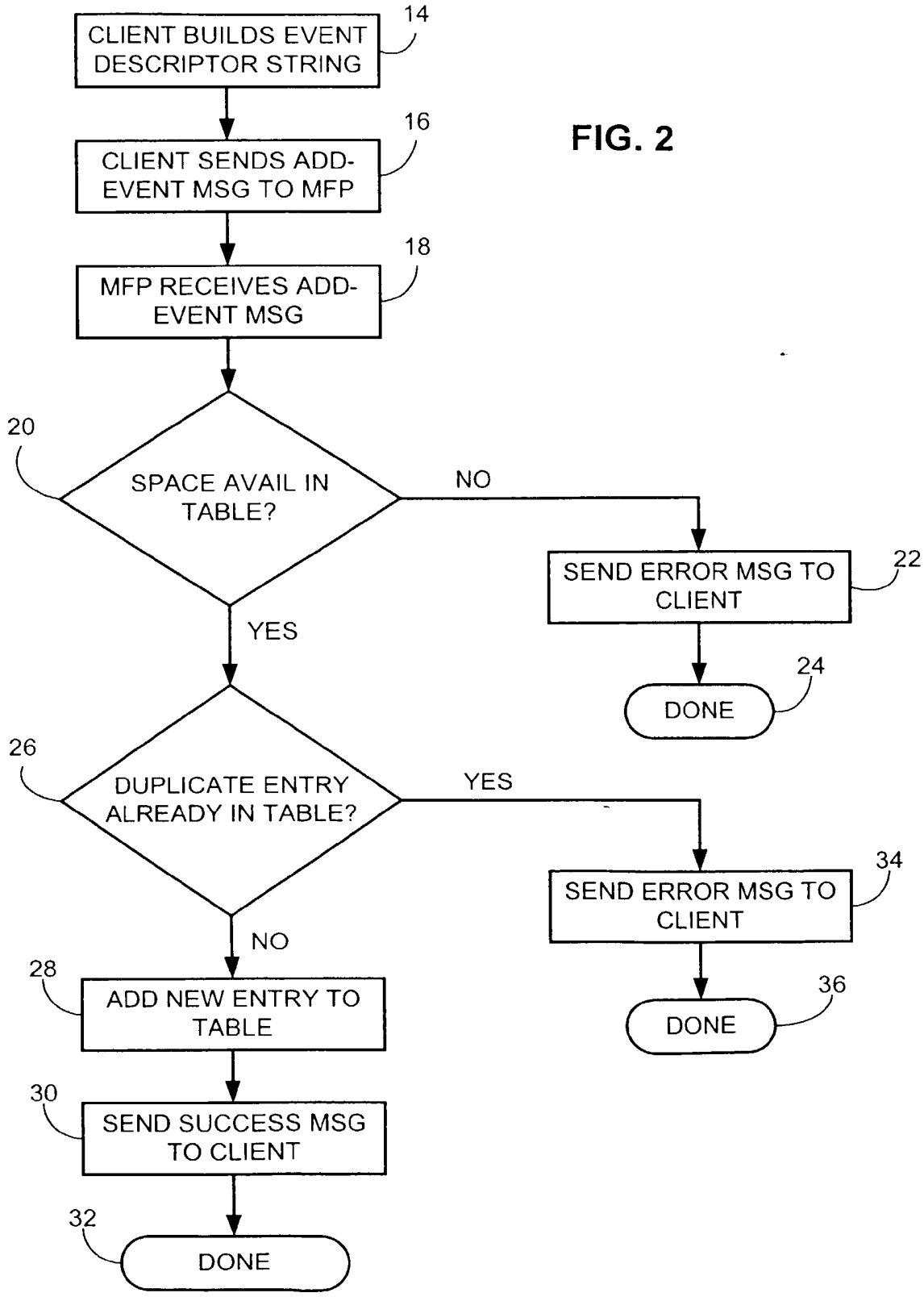

CLIENT BUILDS EVENT DESCRIPTOR STRING
14
CLIENT SENDS ADD-EVENT MSG TO MFP
16
MFP RECEIVES ADD-EVENT MSG
18
20
SPACE AVAIL IN TABLE?
NO
SEND ERROR MSG TO CLIENT
22
DONE
24
YES
26
DUPLICATE ENTRY ALREADY IN TABLE?
YES
SEND ERROR MSG TO CLIENT
34
DONE
36
NO
28
ADD NEW ENTRY TO TABLE
30
SEND SUCCESS MSG TO CLIENT
32
DONE

FIG. 2

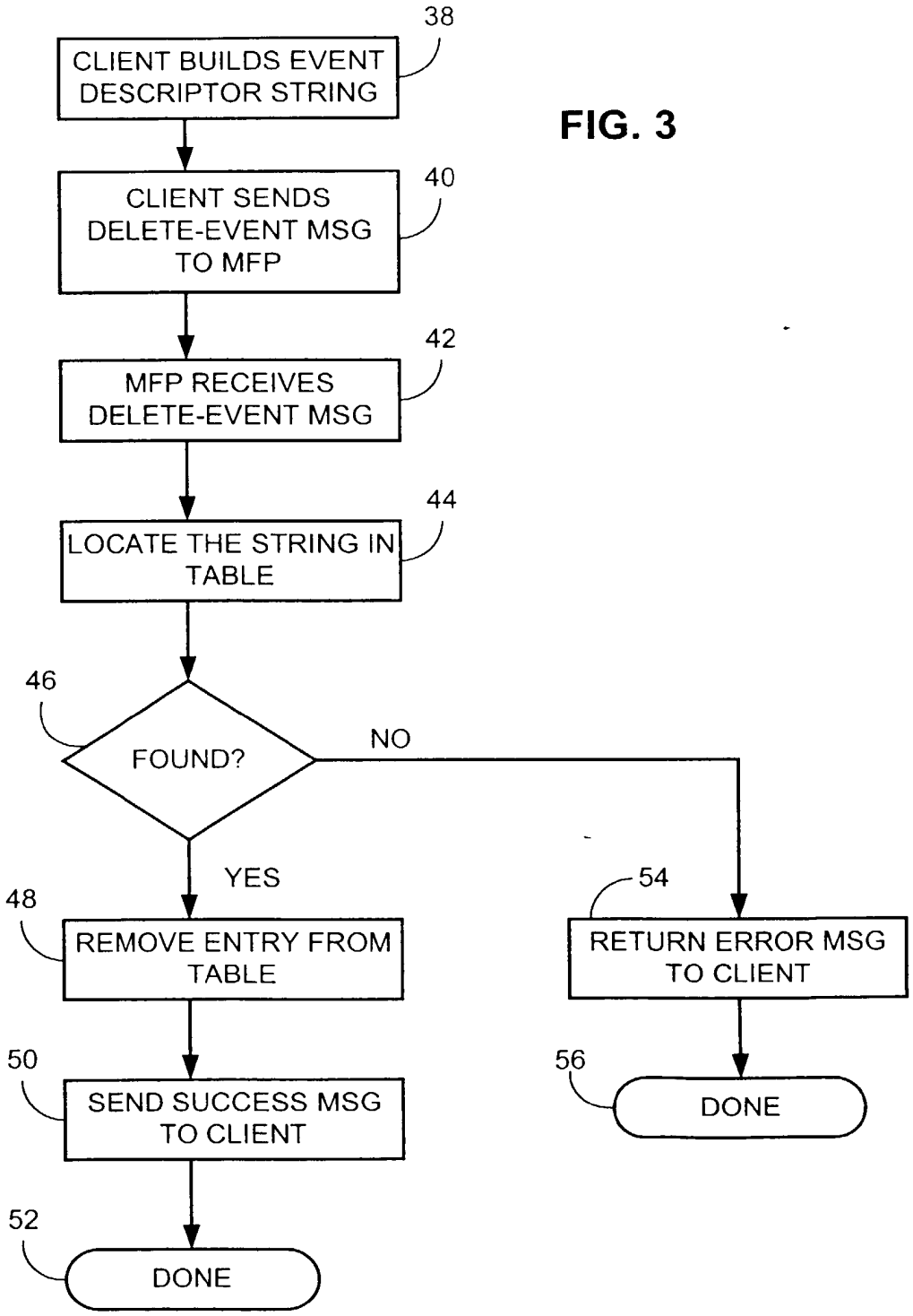

38
CLIENT BUILDS EVENT DESCRIPTOR STRING
40
CLIENT SENDS DELETE-EVENT MSG TO MFP
42
MFP RECEIVES DELETE-EVENT MSG
44
LOCATE THE STRING IN TABLE
46
FOUND?
NO
YES
48
REMOVE ENTRY FROM TABLE
50
SEND SUCCESS MSG TO CLIENT
52
DONE
54
RETURN ERROR MSG TO CLIENT
56
DONE

FIG. 3

FIG. 4

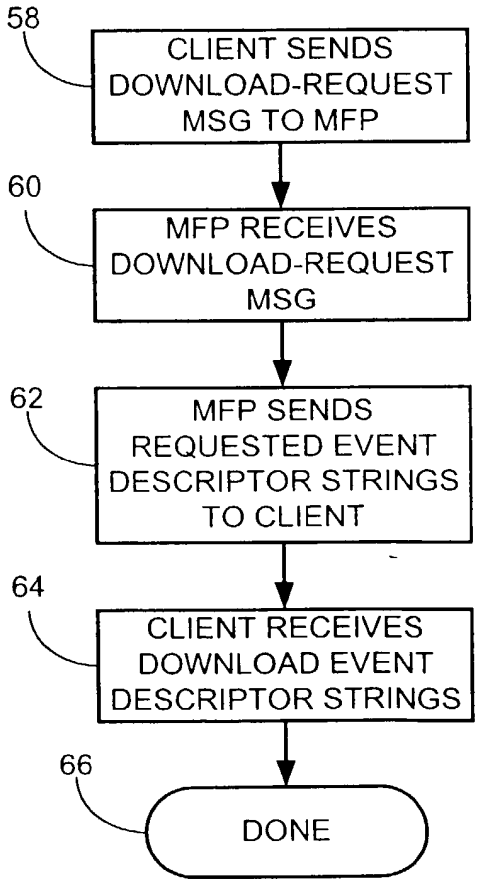
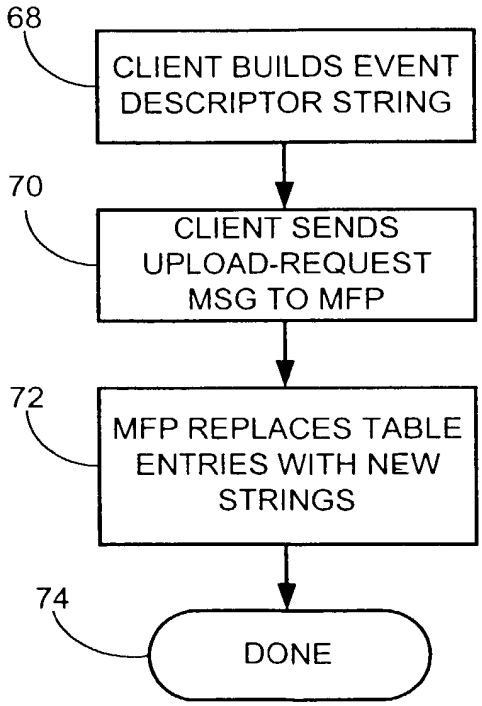

58
CLIENT SENDS DOWNLOAD-REQUEST MSG TO MFP
60
MFP RECEIVES DOWNLOAD-REQUEST MSG
62
MFP SENDS REQUESTED EVENT DESCRIPTOR STRINGS TO CLIENT
64
CLIENT RECEIVES DOWNLOAD EVENT DESCRIPTOR STRINGS
66
DONE

FIG. 5


68
CLIENT BUILDS EVENT DESCRIPTOR STRING
70
CLIENT SENDS UPLOAD-REQUEST MSG TO MFP
72
MFP REPLACES TABLE ENTRIES WITH NEW STRINGS
74
DONE

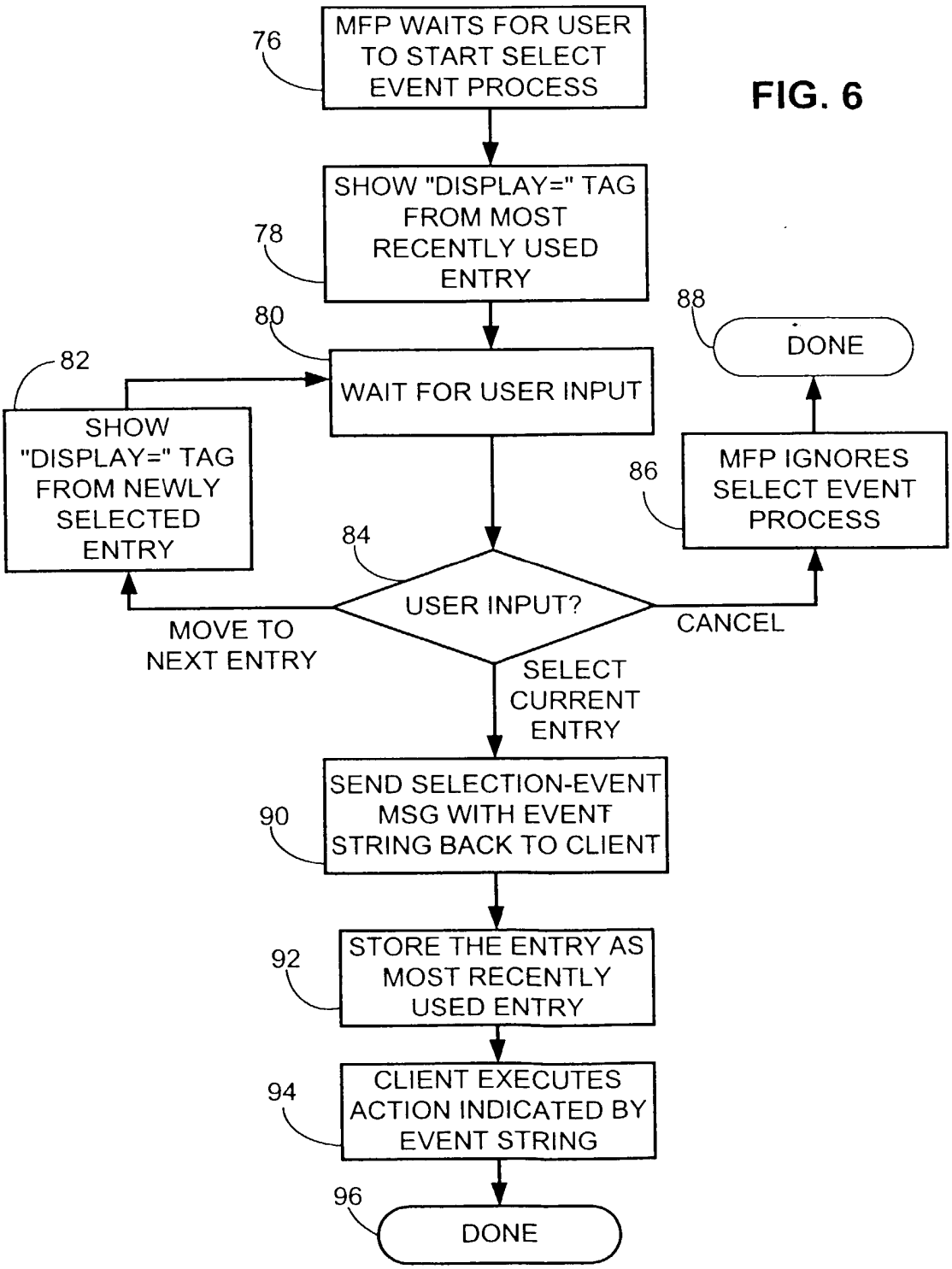

76
MFP WAITS FOR USER TO START SELECT EVENT PROCESS
78
SHOW "DISPLAY=" TAG FROM MOST RECENTLY USED ENTRY
80
WAIT FOR USER INPUT
82
SHOW "DISPLAY=" TAG FROM NEWLY SELECTED ENTRY
84
USER INPUT?
MOVE TO NEXT ENTRY
CANCEL
SELECT CURRENT ENTRY
86
MFP IGNORES SELECT EVENT PROCESS
88
DONE
90
SEND SELECTION-EVENT MSG WITH EVENT STRING BACK TO CLIENT
92
STORE THE ENTRY AS MOST RECENTLY USED ENTRY
94
CLIENT EXECUTES ACTION INDICATED BY EVENT STRING
96
DONE

FIG. 6

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 30 8580

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
Y	EP 0 910 008 A (CANON KK) 21 April 1999 (1999-04-21) * column 22, line 11 - line 58; figures 1,3,4,11,22 * * column 25, line 13 - line 44 * * column 31, line 4 - line 18 *	1,10	G06F13/38
A	* column 34, line 46 - column 35, line 5 *	2-9	
Y	US 5 371 888 A (LEHNERTZ GARY K ET AL) 6 December 1994 (1994-12-06) * column 6, line 30 - line 34; figure 3 *	1,10	
A	EP 0 874 306 A (XEROX CORP) 28 October 1998 (1998-10-28) * page 3, line 50 - line 55; figures 1,7 * * page 6, line 28 - line 29 * * page 8, line 19 - line 26 *	1-10	
A	US 5 220 674 A (BELL GERALD S ET AL) 15 June 1993 (1993-06-15) * column 16, line 39 - column 17, line 20; figure 4 *	1-10	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F
A	"SOFTWARE EVENT SUBSYSTEM" IBM TECHNICAL DISCLOSURE BULLETIN,US,IBM CORP. NEW YORK, vol. 38, no. 12, 1 December 1995 (1995-12-01), pages 509-513, XP000588221 ISSN: 0018-8689 * the whole document *	1-10	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	6 December 2000	Henneman, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 00 30 8580

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

06-12-2000

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0910008 A	21-04-1999	JP 11119925 A	30-04-1999
		JP 11120061 A	30-04-1999
		JP 11120062 A	30-04-1999
		JP 11120133 A	30-04-1999
		JP 11120094 A	30-04-1999
		JP 11119926 A	30-04-1999
		JP 11119956 A	30-04-1999
		JP 11134294 A	21-05-1999
		JP 11134300 A	21-05-1999
US 5371888 A	06-12-1994	JP 2519001 B	31-07-1996
		JP 5216695 A	27-08-1993
EP 0874306 A	28-10-1998	US 5974234 A	26-10-1999
		BR 9801021 A	19-10-1999
		JP 10301732 A	13-11-1998
US 5220674 A	15-06-1993	CA 1307854 A	22-09-1992

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82